ARGUS: SOFTWARE FOR STATISTICAL DISCLOSURE CONTROL OF MICRODATA[1]

A.G. De Waal, A. J. Hundepool and L.C.R.J. Willenborg[2]

ABSTRACT

In recent years Statistics Netherlands has developed a prototype version of a software package, ARGUS, to protect microdata files against statistical disclosure. In 1995 the present prototype version of ARGUS, namely version 1.1, has been released. In this paper both the rules, based on checking low-dimensional combinations of values of so-called identifying variables, and the techniques, global recoding and local suppression, applied by Statistics Netherlands to protect microdata sets are described. Subsequently, it is examined how ARGUS incorporates these ideas. Firstly, the data needed by ARGUS are described. Secondly, it is explained how disclosure control measures can be taken by means of ARGUS. Thirdly, the information generated by ARGUS to determine the global recodings is described. Based on this information the user of ARGUS can decide how the variables should be recoded.

1. INTRODUCTION

To help protect microdata sets against statistical disclosure Statistics Netherlands has developed a prototype version of a software package, ARGUS. In 1995 the present prototype version of ARGUS, ARGUS 1.1, has been released.

ARGUS can handle a class of disclosure control rules based on checking the population frequencies of certain combinations of values of identifying variables. When such a combination does not occur frequently enough in the population suitable disclosure control measures should be applied. ARGUS supports two disclosure control measures, namely global recoding and local suppression. These measures are usually applied simultaneously to a microdata set. In Section 2 the disclosure control rules and measures that can be handled by ARGUS are examined in more detail.

General information on ARGUS is given in Section 3. It is explained what happens when a microdata set is protected by ARGUS. Several steps can be distinguished: the combinations that should be checked according to the rules are generated, the combinations that do not occur frequently enough in the population are determined, the global recodings can be specified interactively by the user, the

[1] The views expressed in this paper are those of the authors and do not necessarily reflect the policies of Statistics Netherlands.

[2] Statistics Netherlands, Department of Statistical Methods, PO Box 4000, 2270 JM Voorburg, The Netherlands (ARGUS@CBS.NL).

local suppressions are determined automatically, and, finally, a report is generated.

To run ARGUS a file with meta information should be constructed. The information that should be included in this file is discussed in Section 4. For example information on the following subjects should be contained in the file: the kind of microdata set that is to be protected, the threshold values to determine whether or not a combination occurs frequently enough in the population, and the estimator that should be used to estimate the population frequency of a combination. Moreover, information on each variable, such as name, should be included in the file with meta information.

To operate ARGUS, commands from five menus can be chosen. Each menu is devoted to one particular task. The five tasks are job handling, e.g. starting and saving a job, obtaining information on variables and combinations, global recoding, local suppression, and generating output, e.g. a report or a protected microdata set. The menus, and their corresponding commands, are described in Section 5.

The two disclosure control measures that are supported by ARGUS, global recoding and local suppressions, differ with respect to the way they are executed. Local suppressions are determined automatically by ARGUS. Global recodings, on the other hand, have to be specified by the user. To help the user to determine suitable global recodings ARGUS provides some information on the 'unsafety' status of variables and combination. This information is described in Section 6.

The paper is concluded by Section 7, in which the future development of ARGUS is discussed. An illustration of an ARGUS-session is given in the Appendix. More information on how to operate ARGUS can be found in De Waal and Pieters (1995) and Pieters and De Waal (1995).

2. SDC FOR MICRODATA AT STATISTICS NETHERLANDS

2.1. Re-identification

The aim of statistical disclosure control is to limit the risk that sensitive information of individual respondents can be disclosed from a data set. In case of a microdata set, i.e. a set of records containing information on individual respondents, such a disclosure of sensitive information of an individual respondent can occur after this respondent has been re-identified. That is, after it has been deduced which record corresponds to this particular individual. So, disclosure control should hamper re-identification of individual respondents.

Re-identification can take place when several values of so-called identifying variables, such as 'Place of residence', 'Sex' and 'Occupation', are taken into consideration. The values of these identifying variables can be assumed known to friends and acquaintances of a respondent. When several values of these identifying variables are combined a respondent may be re-identified. Consider for example the following record obtained from an unknown respondent:

‘Place of residence = Urk’, ‘Sex = Female’ and ‘Occupation = Statistician’.

Urk is a small fishing-village in the Netherlands, in which it is unlikely for many statisticians to live, let alone female ones. So, when we find a statistician in Urk, a female one moreover, in the microdata set, then she is probably the only one. When this is indeed the case, anybody who happens to know this unique female statistician in Urk is able to disclose sensitive information from her record if such information is contained in this record.

An important concept in the theory of re-identification is a *key*. A key is a combination of identifying variables. Keys can be applied to re-identify a respondent. Re-identification of a respondent can occur when this respondent is unique in the population with respect to a certain key value, i.e. a combination of values of identifying variables. Hence, uniqueness of respondents in the population with respect to certain key values should be avoided. When a respondent appears to be unique in the population with respect to a key value, then disclosure control measures should be taken to protect this respondent against re-identification.

In practice, however, it is a bad idea to prevent only the occurrence of respondents in the data file who are unique in the population (with respect to a certain key). For this several reasons can be given. Firstly, there is a practical reason: unicity in the population, in contrast to unicity in the data file, is hard to establish. There is generally no way to determine whether a person who is unique in the data file (with respect to a certain key) is also unique in the population. Secondly, an intruder may use another key than the key(s) considered by the data protector. For instance, the data protector may consider only keys consisting of at most three variables while the intruder may use a key consisting of four variables. Therefore, it is better to avoid the occurrence of combinations of scores that are *rare* in the population in the data file instead of avoiding only population-uniques in the data file. To define what is meant by rare the data protector has to choose a threshold value D_k, for each key value k, where the index k indicates that the threshold value may depend on the key k under consideration. A combination of scores, i.e. a key value, that occurs less than D_k times in the population is considered *rare*, a key value that occurs at least D_k times in the population is considered *common*. Rare combinations of scores that occur in the data file are called *unsafe combinations*. Common combinations of scores, and rare ones that do not occur at all in the microdata set, are called *safe combinations*. The unsafe combinations must be protected, while the safe ones may be published.

There is a practical problem when applying the above rule that the occurrence (in the data file) of combinations of scores that are rare in the population should be avoided. Namely, it is usually not known how often a particular combination of scores occurs in the population. In many cases one only has the data file itself available to *estimate* the frequency of a combination of scores in the population. In practice one therefore uses the estimated frequency of a key value k to determine whether or not this key value is common or not in the population. When the *estimated* frequency of a key value, i.e. a combination of scores, is at least equal to the threshold value D_k, then this combination is considered *common*. When the *estimated* frequency of a key value is less than the threshold value D_k, then this combination is considered *rare*.

2.2. The policy of Statistics Netherlands

Statistics Netherlands has adopted the policy that only certain low-dimensional keys should be examined. In the case of Statistics Netherlands low-dimensional keys means keys consisting of at most three variables. An example of such a low-dimensional key is the following one: 'Place of residence' ×'Sex' ×'Occupation'. Although Statistics Netherlands restricts itself to three-dimensional keys, ARGUS is able to deal with keys up to five dimensions. A reason why Statistics Netherlands considers low-dimensional keys only is that it releases two specific kinds of microdata files. The first kind of microdata files are so-called *public use files*. According to the rules of Statistics Netherlands these files may not contain any sensitive information. Therefore, it is not necessary to pay much attention to the identifying variables, i.e. because public use files do not contain sensitive information it is sufficient to examine only low-dimensional keys. The second kind of microdata files are *microdata files for research*. These files are only released to trustworthy researchers, who, moreover, have to sign a contract stating that they will not misuse the data file to disclose information about individual respondents. Because of this contract and the fact the data files for research are released to a select group of trusted researchers only, these data files are not severely protected. In fact, they are protected to avoid so-called *spontaneous recognition* only. That is, it is assumed that the users of a microdata set for research will not make a deliberate attempt to disclose sensitive information of individual respondents. However, they may recognize a respondent spontaneously when examining (low-dimensional) tables. This implies that the keys that have to be examined are low-dimensional ones.

2.3. SDC techniques

Statistics Netherlands advocates two SDC techniques to protect microdata sets, namely global recoding and local suppression. In case of global recoding several categories of a variable are collapsed into a single one. In the above example, for instance, we can recode the variable 'Occupation'. For instance, the categories 'Statistician' and 'Mathematician' can be combined into a single category 'Statistician or Mathematician'. When the number of female statisticians in Urk plus the number of female mathematicians in Urk is sufficiently high, then the combination 'Place of residence = Urk', 'Sex = Female' and 'Occupation = Statistician or Mathematician' is considered safe for release. Note that instead of recoding 'Occupation' one could also recode 'Place of residence' for instance.

It is important to realize that global recoding is applied to the whole data set, not only to the unsafe part of the set. This is done to obtain an uniform categorization of each variable. Suppose, for instance, that we recode the 'Occupation' in the above way. Suppose furthermore that both the combinations 'Place of residence = Amsterdam', 'Sex = Female' and 'Occupation = Statistician', and 'Place of residence = Amsterdam’, ‘Sex = Female’ and ‘Occupation = Mathematician’ are considered safe. To obtain a uniform categorization of ‘Occupation’ we would, however, not publish these combinations, but only the combination ‘Place of residence = Amsterdam’, ‘Sex = Female’ and ‘Occupation = Statistician or Mathematician’.

When local suppression is applied one or more values in an unsafe combination are suppressed, i.e. replaced by a missing value. For instance, in the above example we can protect the unsafe combination ‘Place of residence = Urk’, ‘Sex = Female’ and ‘Occupation = Statistician’ by suppressing the value of ‘Occupation’, assuming that the number of females in Urk is sufficiently high. The resulting combination is then given by ‘Place of residence = Urk’, ‘Sex = Female’ and ‘Occupation = missing’. Note that instead of suppressing the value of ‘Occupation’ one could also suppress the value of another variable of the unsafe combination. For instance, when the number of female statisticians in the Netherlands is sufficiently high then one could suppress the value of ‘Place of residence’ instead of the value of ‘Occupation’ in the above example to protect the unsafe combination. A local suppression is only applied to a particular value. When, for instance, the value of ‘Occupation’ is suppressed in a particular record, then this does not imply that the value of ‘Occupation’ has to be suppressed in another record. The freedom that one has in selecting the values that are to be suppressed allows one to minimize the number of local suppressions. More on this subject can be found in De Waal and Willenborg (1995).

Both global recoding and local suppression lead to a loss of information, because either less detailed information is provided or some information is not given at all. A balance between global recoding and local suppression has to be found in order to make the information loss due to SDC measures as low as possible. Statistics Netherlands recommends to start by recoding some variables globally until the number of unsafe combinations that have to be protected by local suppression is sufficiently low. The remaining unsafe combinations have to be protected by suppressing some values.

ARGUS allows the user to specify the global recodings interactively. Some information on how to choose these global recodings is provided by ARGUS. In case the user is not satisfied with a particular global recoding it is easy to undo it. After the global recodings have been specified the values that have to be suppressed are determined automatically and optimally, i.e. the number of values that have to be suppressed is minimized. This latter aspect of ARGUS, determining the necessary local suppressions automatically and optimally, makes it possible to protect a microdata set against disclosure quickly.

3. GENERAL INFORMATION ON ARGUS

As is explained Section 2 a microdata file should be protected against disclosure in two steps according to the policy Statistics Netherlands has currently adopted. In the first step some variables should be globally recoded. In the second step some values of variables should be locally suppressed. ARGUS has been developed to perform the necessary tasks. In this section the way ARGUS performs these tasks is sketched.

To run ARGUS one system file or two ASCII files are necessary. When there is no system file available a new job has to be started. In that case two ASCII files should be present, namely one containing the data set that is to be protected and one containing certain meta information. This latter file is called the meta file. More information about the meta file can be found in Section 4.

When a new job is started ARGUS starts by reading the values of all identifying variables for all records into the computer's memory. At the moment there may be at most 50 identifying variables. The values of the non-identifying variables are not read into the computer's memory. Based on the information in the meta file ARGUS then generates a set of combinations that have to be examined. Each combination may consist of values of at most 5 variables. Subsequently, all the combinations generated are indeed examined, i.e. the estimated population frequency of each combination is compared with the corresponding threshold value. A database with information on the variables and the unsafe combinations is then constructed. This database is called the *combinations database*.

After construction of this database the user can specify the global recodings. To help the user determine these global recodings ARGUS provides some information. When all global recodings have been interactively specified by the user, ARGUS determines the remaining necessary local suppressions automatically and optimally, i.e. the number of local suppressions is minimized. Finally, ARGUS generates a report and creates the protected data file.

Something that may be surprising is that a protected, safe, data file created by ARGUS may seem to be unsafe when presented again to ARGUS! This is a consequence of suppressing values. A combination of scores is considered common when the estimated frequency of that combination in the population is at least equal to a certain threshold. These estimates are based on the data file that is to be protected itself. Now suppose that a particular combination of scores seems to occur frequently enough when its estimated frequency is based on the original unprotected data file. Suppose furthermore that due to suppressions this combination occurs less often in the protected data file then in the unprotected one. In this case it may happen that the estimated frequency of this combination in the population based on the protected data file is less than the threshold value. So, based on the protected data file we would conclude that this combination is unsafe. This is not correct though, because we know from the unprotected data file that this combination is safe. That it seems to be unsafe when the protected data file is considered only, is a consequence of the fact that some values have been suppressed which leads to a lower estimate for the population frequency of the combination under consideration.

4. DESCRIPTION OF THE META FILE

In this section we examine the meta file that is necessary for an ARGUS-session. We begin by describing the two kinds of files, MICROFILE and PUBLICFILE, one can protect by means of ARGUS. These kinds of files differ with respect to the combinations that are generated and subsequently checked. Then we describe the criteria that can be used to check whether a combination that is generated by ARGUS is safe or not. Two criteria can be chosen: an absolute criterion and a relative criterion. How to prescribe the corresponding threshold values is described next. As ARGUS uses *estimated* frequencies to determine whether or not a combination is safe, one has to indicate which estimator one wishes to apply. There are two kinds of estimators that are supported by ARGUS: an interval estimator and a compromise estimator. The compromise estimator can be applied with or without auxiliary information. We explain how one can select one of these estimators. Local suppressions are determined automatically and optimally. In practice, one does not want too many local suppressions, however, as this would introduce bias in the statistical results obtained from the protected microdata set. This can be accomplished by specifying an upper bound for the number of local suppressions. We describe how this upper bound can be fixed. Finally, we describe the information that has to be provided for each variable in the microdata set.

Kind of file:

In the meta file it has to indicated whether a MICROFILE, i.e. a microdata file for research, or a PUBLICFILE, i.e. a public use file, is to be protected. In the case of Statistics Netherlands usually a MICROFILE has to be protected. There are some differences between protecting a MICROFILE and protecting a PUBLICFILE. The most important difference is that the combinations that are generated by ARGUS and that subsequently have to be checked are different for the two kinds of files. When a MICROFILE is to be protected all combinations of variables with a different identification level are generated, when a PUBLICFILE is to be protected all uni- and bivariate combinations of variables with a non-zero identification level are generated. Another difference is explained in the subsection ‘Threshold values’ below.

Kind of criterion:

The kind of criterion that should be applied can be selected by the command ABSOLUTE_CRITERION or RELATIVE_CRITERION. Both of these commands should be followed by three numbers d_1, d_2 and g, where $d_2 \geq d_1$ and g lies between zero and one. These numbers are used only when a compromise estimator is applied to estimate the population frequencies. They are in fact population threshold values to determine whether a combination is common or rare. When ABSOLUTE_CRITERION is used only d_1 is important. When the estimated population frequency of a combination is at least equal to d_1 then this combination is considered common, otherwise it is considered rare. This criterion is called an absolute criterion, because only absolute frequencies are taken into consideration. RELATIVE_CRITERION is more complicated. In this case ARGUS starts by checking whether the estimated fraction of elements with a certain combination of scores in the population is at least equal to g. If this is the case, then ARGUS checks whether the estimated population frequency of elements with this combination of scores is at least equal to d_1. If the estimated population frequency is indeed at least equal to this number, then the combination is considered common, otherwise it is considered rare. When on the other hand the estimated fraction of elements with this combination of scores is less than g, then ARGUS checks whether the estimated population frequency is at least equal to d_2. If the estimated frequency is indeed at least equal to d_2, then the combination is considered common, otherwise it is considered rare. This criterion is called a relative criterion, because in the first step it considers the relative fraction of a combination of scores. Depending on this fraction the absolute threshold value changes. For more information on the absolute criterion and the relative criterion see De Vries, De Waal and Willenborg (1994).

Threshold values:

A number of threshold values should be specified after the SAFELIMITS command. These threshold values are used only when the so-called DEFAULT_ESTIMATOR is applied. For a MICROFILE the threshold value for the combinations should be specified. A combination is considered common when its frequency in the data file is at least equal to this threshold value, otherwise it is considered rare. For a PUBLICFILE we have two threshold values: one for the univariate 'combinations' and the other for the bivariate combinations. In this case, the first number in the meta file indicates the threshold value for the univariate 'combinations' and the second number the threshold value for the bivariate combinations. The number(s) given after SAFELIMITS are threshold values for the data file and not population threshold values. At Statistics Netherlands population threshold values are first translated to threshold values for the data file. This translation has to be done outside ARGUS. To translate population frequencies to sample frequencies it is assumed that the frequency of a combination in the data file is Poisson distributed, and by testing whether the frequency of a combination is higher than the corresponding population threshold value. More information on this approach can be found in Pannekoek (1989).

Kind of estimator:

DEFAULT_ESTIMATOR indicates that the SAFELIMITS are used. Instead of using the SAFELIMITS, one can also apply two compromise estimators. In that case one should type COMBI_ESTIMATOR followed by a '1' for the first compromise estimator or a '2' for the second estimator. In this case the population threshold values given by the 'Kind of criterion' command are used (see subsection 'Kind of criterion' above). The first compromise estimator does not use auxiliary information, whereas the second one does. When one wants to use a compromise estimator then there should be exactly one regional variable in the data file. Moreover, an additional data file should be provided. When the first compromise estimator is used, i.e. the compromise estimator without auxiliary information, then each line of this additional data file should contain a value of the regional variable followed by the number of elements, e.g. persons, in the target population in this region. Each value of the regional variable should occur exactly once in this data file. When the second compromise estimator is used, i.e.

the compromise estimator with auxiliary information, then each line of the additional data file should contain a value of the regional variable, the number of persons in the target population that this value refers to and the value of the auxiliary variable. When the regional variable is the place of residence of the respondents then one could use the degree of urbanization as auxiliary variable for instance. More information on these compromise estimators can be found in Pannekoek and De Waal (1994).

Maximum number of local suppressions:

The maximum number of values that may be suppressed by ARGUS should be indicated. When the minimal number of local suppressions found by ARGUS exceeds this maximum number ARGUS returns an error message.

Variable information:

We now describe the information that has to be specified for each variable in the meta file.

1. Variable name

The name of each variable given and its (fixed) place in the data file should be specified.

2. Missing values

For each variable two missing values have to be specified. One of these missing values is used when the respondent did not have to answer the corresponding question due to the routing structure of the questionnaire, the other missing value is used when the respondent refused to answer this question.

3. Identification level

The *identification level* of each variable has to be described. The more identifying the variable is considered to be, the lower the identification level should be chosen. So, identification level 1 is used for variables that are considered very identifying, identification level 2 is used for variables that are somewhat less identifying, and so on. The highest identification level that may be assigned to a variable is 5. When protecting a so-called MICROFILE, it is important to realize that variables with a low identification level are considered to be a subset of variables with a higher identification level. For instance, variables with identification level 1 are considered to be a subset of the variables with identification level 2. Likewise, the variables with identification level 2 are considered to be a subset of variables with identification level 3. The combinations that are generated by ARGUS in this case are all combinations of variables with different identification levels, where the above hierarchical structure is taken into account. The number of variables in such a combination is at most equal to the highest identification level assigned to a variable. When on the other hand a so-called PUBLICFILE is to be protected, then all uni- and bivariate combinations of variables with a non-zero identification level are generated by ARGUS. The combinations that have been generated are subsequently checked.

To illustrate which combinations are generated in case of a MICROFILE we assume that there are variables of three different identification levels, 1, 2, and 3. Variables of each level are examined univariately. Bivariately, the combinations ‘level 1 variable’×‘level 2 variable’, ‘level 1 variable’×‘level 3 variable’, and ‘level 2 variable’×‘level 3 variable’ are generated. Here the hierarchical structure of the variables is taken into account by ARGUS. For instance, each level 1 variable is also a level 2 variable so combinations ‘level 1 variable’×‘level 1 variable’ are generated by ARGUS. Likewise, ARGUS generates the combinations ‘level 2 variable’×‘level 2 variable’. Finally, the three-dimensional combinations ‘level 1 variable’×‘level 2 variable’×‘level 3 variable’ are generated. Again, the hierarchical structure of the variables is taken into account. So, in fact the combinations ‘level 1 variable’×‘level 2 variable’×‘level 2 variable’, ‘level 1 variable’×‘level 1 variable’×‘level 3 variable’, ‘level 1 variable’×‘level 1 variable’×‘level 2 variable’, and ‘level 1 variable’×‘level 1 variable’×‘level 1 variable’ are also generated by ARGUS.

4. Priority number

A *priority number* has to be specified for each variable. When ARGUS suppresses values it not only minimizes the total number of suppressions, but it also tries to suppress values of variables with low priority numbers. Technically, ARGUS first examines whether it can minimize the total number of suppressions by suppressing values of variables with low priority numbers. When this fails, ARGUS also considers suppression of values of variables with a higher priority number. This implies that when the priority numbers have been badly chosen, i.e. when the total number of suppressions can only be minimized when values of variables with high priority numbers are suppressed, ARGUS will need much time to find the optimal solution.

5. Compromise estimation

To use one of the two compromise estimators that are available in ARGUS it is necessary to indicate which variables are *regional variables*, i.e. variables of which the values refer to regions. Whether or not a variable is a regional variable is important only when a *compromise estimator* is applied to estimate population frequencies.

6. Hierarchical variables

Some variables are hierarchical variables. Each digit of such a variable corresponds to a particular hierarchical level. These variables can be recoded in a special way, namely by truncation. This means that some trailing digits from the values of these variables may be deleted. Whether or not a variable may be truncated has to be indicated.

5. OPERATING ARGUS

In this section we examine how ARGUS can be operated once the data file and meta file have been constructed. ARGUS can be operated by choosing commands interactively from several menus. In fact, ARGUS provides five menus, namely one for job handling, one for information on variables and combinations, one for global recoding, one for local suppression and one for producing output.

5.1 Handling jobs

From the menu for job handling a new job can be started at any moment. When the user was working on another job already, ARGUS asks whether that job should be saved or not. It is also possible to go on with an existing job. The protection measures that have been applied to the corresponding data file during the previous session are then automatically executed. To be able to go on with an existing job this job must have been saved during a previous ARGUS-session. When a job is saved ARGUS in fact saves the corresponding system file.

5.2 Obtaining information on variables and combinations

From the menu for information on variables and combinations frequency count tables and information on the ‘unsafety’ status of combinations or variables can be examined. Information on the ‘unsafety’ status of combinations or variables can be obtained from the combinations database. One can also examine frequency count tables. The frequency count table that one wants to see can be specified in two ways. Firstly, the frequency count table can be selected from the combinations that have been generated by ARGUS. Secondly, the frequency count table can be specified manually. In both cases the resulting frequency count table can be viewed. A combination from this list can be selected.

Each line of a frequency count table contains the cell number, the values of the variable(s), and the number of records that score on that particular combination of values. A univariate frequency count table contains the value of *#UCells* in addition. This value provides some information on how to choose the global recodings. The precise meaning of *#UCells* is explained in Section 6.

5.3 Global recoding

To recode a variable globally the global recoding menu should be used. When recoding variables globally it is handy to obtain information about the variables quickly. The user can see information on the ‘unsafety’ of the variables. The information that can be obtained is contained in a list of variables that can be shown on the screen. This list contains the name of the variable, the value of *#UCells*, the value of *#UCat*, and the value of *#UComb*. These values provide information on how to choose the global recodings. The precise meaning of *#UCells*, *#UCat* and *#UComb* is again explained in Section 6. The univariate frequency count table of a variable can be shown on the screen by selecting one of the variables. This table contains the cell number, the values of the variable, the frequencies of these values and the values of *#UCells*.

Now we describe how a global recoding can be specified. Recoding variables globally is usually done in two steps. Firstly, a file containing information on how to recode a particular variable should be made. This file is called the *recoding file*. Secondly, this global recoding should be performed. Constructing a recoding file can be done during an ARGUS-session. For the variable that is selected a recoding file will be made.

When a recoding is specified several checks are made. For instance, it is checked whether the new categories lie within the range specified in the meta file and whether all old values are recoded at most once. Values that are not affected by the recoding do not have to be listed. Because a recoding file is an ASCII file it can be prepared before an ARGUS-session. A recoding file can also be used several times. This is especially useful when one wants to apply certain standard recodings. These standard recodings could be applied to the same variable, such as ‘Place of residence’, in different data sets, or to similar variables, such as ‘Place of residence’ and ‘Place of work’, in a single data set. After a recoding file has been produced the corresponding recoding can be executed. A list of the variables can be shown on the screen. By selecting one of these variables the corresponding recoding is performed.

Variables can be truncated. From a list of variables a variable that may be truncated should be chosen. When such a variable is chosen ARGUS poses the question how many trailing digits should be deleted. If the answer is ‘1’, for instance, then the last digit of each value is deleted. A variable may be recoded by truncating it and subsequently recoding it by means of a recoding file, or vice versa.

When a recoding is not satisfying then this recoding can be undone easily. A list of variables can again be shown on the screen. By choosing one of the variables in this list all recodings, truncations as well as recodings specified by the recoding file, applied to this variable are undone.

5.4 Local suppression

After the global recodings have been specified the local recodings have to be determined. This can be done by means of the local suppression menu. ARGUS determines the local suppressions automatically. In some cases, however, one wants to exclude some variable from local suppression. Therefore, ARGUS allows one to exclude values of certain variables from suppression. Excluding variables from suppression should be done with care. In case the variables to be excluded from local suppression are badly chosen it may be impossible to protect the data file. For instance, when the number of female statisticians in Urk is too low and all three variables ‘Place of residence’, ‘Sex’ and ‘Occupation’ are excluded from suppression then it is clearly impossible to protect the combination ‘Place of residence = Urk’, ‘Sex = Female’ and ‘Occupation = Statistician’ by means of suppression.

The local suppressions are determined automatically and optimally, i.e. the number of local suppressions is minimized. The resulting data file does not contain any unsafe combinations. When one is not satisfied with the (number of) local suppressions it is possible to undo them. After undoing the local suppressions one can try some other global recodings and then determine the local suppressions again.

5.5 Producing output

After the global recodings and local suppressions have been determined a report on the SDC measures taken can be generated. Such a report can be generated from the menu for producing output. This report contains information on the applied global recodings, including the truncations, and the values that have been suppressed locally. After a report has been generated it can be shown on the screen. This has the advantage that if the result is not satisfying then other global recodings and local suppressions can be tried immediately, i.e. without leaving ARGUS first.

The recoded, protected, file can be generated. This protected data file contains all variables from the original, unprotected, data file. In other words, the protected data file does not only contain the identifying variables, but also the non-identifying ones. The records of the recoded file can be placed in random order. Randomizing data files helps to increase the safety of data files. The reason for randomizing a data file is that data are often stored in a specific order, for instance all respondents from the same household or the same region are grouped together. Such a grouping increases the disclosure risk. Randomizing the data file destroys this grouping.

6. RECODING INFORMATION GENERATED BY ARGUS

To help the user determine the global recodings ARGUS provides some information on the 'unsafety' of variables and combinations. In particular, information on *#UCells*, *#UCat* and *#UComb* can be shown on the screen. The value *#UCells* of a variable equals the number of times that this variable occurs in an unsafe cell, i.e. an unsafe combination of values of variables. The value *#UCat* of a variable equals the number of categories of this variable that occur in unsafe combinations. Finally, the value *#UComb* of a variable is the number of times that this variable occurs in an unsafe combination of variables. When the numbers *#UCells*, *#UCat* and *#UComb* are high for a certain variable then this variable is rather unsafe, and should be recoded.

Note that when *#UCells* equals, say, 516 this does not imply that 516 values will be suppressed when automatic and optimal suppression is applied, because several unsafe combinations may be protected simultaneously by suppressing a single value in case these unsafe combinations occur in the same record. However, the total number of unsafe cells, 516 in this case, is an upper bound for the number of necessary local suppressions. When the total number of unsafe cells is low, the number of local suppressions will be low as well. This kind of information can be useful when deciding whether or not additional global recodings should be made.

Information on the 'unsafety' of categories, i.e. values of variables, can also be shown on the screen. This information consists of the value of *#UCells* for each category of a variable, i.e. the number of times that this category occurs in an unsafe combination. When the value of *#UCells* is high the corresponding category is rather unsafe, and should be recoded.

7. FUTURE DEVELOPMENT OF ARGUS

At the moment ARGUS is rapidly being developed further. In fact, this further development of ARGUS is a major aim of an ESPRIT-project on SDC. The present prototype will be extended and improved upon. This will result in a package, μ-ARGUS, for the SDC of microdata. Apart from this package for microdata another package for the SDC of tabular data, τ-ARGUS, will be developed.

In contrast to the development of ARGUS 1.1 the development of these two packages is not undertaken by Statistics Netherlands alone. Besides Statistics Netherlands six scientific and statistical institutions are actively involved in the ESPRIT-project. These participating institutions are Eindhoven University of Technology, the University of Manchester, the University of Leeds, the Office of Population Censuses and Surveys (OPCS), the Istituto Nazionale di Statistica (ISTAT), the Consortio Padova Ricerche (CPR), and Statistics Netherlands, acting as project coordinator. The international cooperation on the further development of ARGUS hopefully will not only lead to a more powerful package for SDC, but also to a harmonization of SDC-rules.

APPENDIX

USING ARGUS, AN EXAMPLE

In this section we will give an impression of the use of ARGUS to protect a micro-data file against disclosure risks. We will apply the rules that have been established at Statistics Netherlands for the generation of a safe micro-file. ARGUS however is capable of applying a much wider class of rules. The file produced according to these rules is considered suitable for the use by people at universities and other research institutes, that have a contract with Statistics Netherlands to use such files only for analysis purposes.

WHAT CAN ARGUS DO

According to the rules we must inspect a large set of tables (univariate, bivariate and trivariate cross-tabulations) to find unsafe combinations. Unsafe will mean that this combination is 'rare' in the population and therefore a risk of disclosure. These combinations can then be made safe by recoding some variables (making them less detailed) or by changing some of the codes in a record into the category 'missing'.

An ARGUS job will read the data file and store it efficiently in the memory of the computer. Then it will generate all the tables to be inspected and help you to select and apply appropriate global recodings. These global recodings will do the major part of the disclosure protection job. After the global recodings most of the combinations will be safe, but any remaining unsafe combinations will then be treated by local suppression. Some variables in the remaining records will be recoded into 'missing'.

When this the file is safe it can be stored on disk again. Also a report is generated describing all the steps taken during this process. You will see that even a rather large data set can be protected against disclosure risks on a moderate computer. This demo was run on a 486/33Mhz computer with 8Mb RAM connected to a Novell file server containing the software and the data file.

THE EXAMPLE

In this description we will gave an impression of a typical ARGUS run, without elaborating on all the details. In this example we will use a file containing 89 498 records from a month's portion of the continuous Labor Force Survey in the Netherlands. The most important (as far as disclosure risks are concerned) variables are 'Region', 'Gender', 'Marital status', 'Age', 'Ethnicity', 'Labor-market-position', and 'Education'.

According to the rules of Statistics Netherlands the variables have to be divided into 4 categories: more identifying variables, more identifying ones, identifying ones, and other variables. A most identifying variable is also a more identifying one. Similarly, a more identifying variable is also an identifying one. The combinations given a value of a most identifying variable by a value of a more identifying variable by a value of an identifying variable have to be checked. These combinations have to occur sufficiently often in the population to be considered safe.

The classification of the variables in the continuous Labor Force Survey into the 4 categories is as follows:

1. most identifying variables

    Region (28 cat.)

2. more identifying variables , but not most identifying variables

   Gender (2 cat.)

   Ethnicity (27 cat.)

3. identifying variables, but not more identifying variables

   Age (60 cat.)

   Marital status (4 cat.)

   Position in household (9 cat.)

   Number of children (7 cat.)

   Education level (784 cat.)

   Labor market position (10 cat.)

   Occupation (842 cat.)

   Type of enterprise (321 cat.)

   and some other variables

4. other variables

From prior experiences it was already quite clear that in data files like this one the variable ‘Region’ (± 700 cat.) is by no means allowed, so we took a higher level of aggregation (28 cat.) from the beginning. Using the original codes for ‘Region’ would be very well possible but 32 Mb RAM of memory on our computer would have been required.

The rules in the Netherlands speak about 3 levels of identification, but ARGUS allows for up to 5 different levels of identification.

PREPARATION OF THE META DATA

Each package that is used to process a statistical data file must have some knowledge about the structure of the data file (i.e. the meta data). If this data file would have been created directly with Blaise, we could have transformed most of the meta data automatically from the Blaise questionnaire. In this case the Blaise questionnaire was not available for us so we had to specify the meta data ourselves.

ARGUS expects the data in a fixed format ASCII file. For each variable we need to specify the name, starting position, length, missing values (2 are possible), the identification level, a priority level for local suppression, and an indication whether the coding scheme for this variable is a hierarchical one. In this case the removal of the last digit is a permitted recoding operation. For one variable this would look like:

```
Ethnicity   36    2   99  99  2  3  0
```

(starting position = 36, length = 2, missing values = 99 and 99 (i.e. in fact only one missing value), identification level = 2, suppression priority = 3 and 0 indicates no hierarchical code).

Only variables with an identification level greater than 0 need to be specified. All the other variables however will be copied into the resulting safe file when this file is created at the end of the job. It is not necessary to specify the code lists of the variables. ARGUS will create them itself, with the restriction that the codes are strictly numerical.

Some general settings that have to be specified are:

a public use file or a micro data file under contract is required,

the file contains leading spaces or leading zeros the specification of your favorite editor, used amongst other things to specify or modify recoding schemes

the record length

- the maximum number of local suppressions allowed
- the minimum cell value considered safe

THE FIRST STEP

When the meta data has been prepared we can start ARGUS. The first step is loading the data file into memory. In the first round ARGUS will read the data file to establish the code lists of the variables considered. In the second round ARGUS will actually read the variables into the memory. This took 2 minutes 47 seconds. The next step is to calculate all the tables. The list of tables results from the different identification levels of the variables as specified in the meta data. This list contains 163 tables. The generation of these tables took 1 minute 28 seconds.

We are now ready to inspect the available information to decide which global recodings are necessary.

When we look at the most striking bivariate tables we get the following result:

Table			Unsafe Combinations
Region	×	Occupation	4270
Region	×	Education	3474
Ethnicity	×	Occupation	1218
Ethnicity	×	Education	1199
Region	×	Enterprise	1198
Ethnicity	×	Enterprise	769
Ethnicity	×	Age	252

This information suggests that the first step would be to do some recoding for ‘Occupation’, ‘Education’ and ‘Enterprise’. These three variables are all hierarchical, so the last digit can be removed for all three variables. Also ‘Ethnicity’ and ‘Region’ are candidates for global recoding but we will treat them later and concentrate on the other ones first.

The results after this recoding are much better now but still there are too many unsafe combinations:

Table			Unsafe Combinations
Region	×	Occupation	1323
Region	×	Education	1810
Ethnicity	×	Occupation	692
Ethnicity	×	Education	812
Region	×	Enterprise	110
Ethnicity	×	Enterprise	210
Ethnicity	×	Age	252

The next step is to recode ‘Ethnicity’ into 3 categories (Dutch, European and other) and to recode the region to the level of provinces (12). These recoding schemes are entered with a simple text editor by specifying which original codes are to be combined into a single new code. However in most cases these recoding schemes are already available e.g. from some coordination department. These two recodings improve the results again.

Table			Unsafe Combinations
Region	×	Occupation	483
Region	×	Education	696
Ethnicity	×	Occupation	126
Ethnicity	×	Education	177
Region	×	Enterprise	44
Ethnicity	×	Enterprise	7
Ethnicity	×	Age	0

A final step will be to further recode ‘Occupation’ and ‘Education’. We will remove another digit. This looks more dramatic than it really is. In total the number of different categories is reduced from 842 to 88 for occupation and from 784 to 119 for ‘Education’. The results are now:

Table			Unsafe Combinations
Region	×	Occupation	60
Region	×	Education	164
Ethnicity	×	Occupation	14
Ethnicity	×	Education	7
Region	×	Enterprise	44
Ethnicity	×	Enterprise	7
Ethnicity	×	Age	0

After these three rounds of global recoding we decide that the rest will be taken care of local suppressions. Due to the implementation of global recodings in ARGUS these are executed very quickly. One global recoding takes only one second, while the recalculation of the set of base tables takes about 1 minute. So ARGUS offers you a fast-working tool (even on a larger data set) to play with the various recoding possibilities. A simple ‘undo’ function is available if you want to investigate some alternatives.

LOCAL SUPPRESSIONS

When you have decided to switch to local suppressions to deal with the last remaining unsafe combinations, ARGUS will do it for you without any further interaction from the user. This step is more time consuming as ARGUS will try to find a optimal suppression in all the unsafe records, i.e. with a minimum number of suppressions. In our case it took about 10 minutes. But after that you are sure that you have created a ‘safe’ file according to the ‘rules’.

In total we have the following result for the local suppression:

Total number of suppressions: 1778

Variable	Number of suppressions
Region	118
Gender	413
Ethnicity	1049
Education	178
Occupation	3
Enterprise	4

FINAL STEP

The final step is to generate a report and to write the protected file to disk. In the report an overview is given of the actions of the ARGUS session. All the global recodings are listed and also all the local suppressions. This can be used to document the ARGUS session.

CONCLUSION

The above gives you an impression of a typical ARGUS session with the current prototype of ARGUS. Even a moderately large data set can be protected against possible disclosure risks on an average PC fairly quickly. Further developments will include amongst other things a revision of the interface (Windows), allowing really large data sets and the inclusion of optimization routines to find a good set of global recodings and local suppressions.

A demonstration disk and further information can be obtained by contacting the authors at Statistics Netherlands (e-mail: ARGUS@CBS.NL).

REFERENCES

De Vries, R.E., A.G. de Waal and L.C.R.J. Willenborg, 1994, Distinguishing rare from common characteristics in a microdata set. Report, Statistics Netherlands.

De Waal, A.G. and A.J. Pieters, 1995, ARGUS User’s guide. Report, Statistics Netherlands, Voorburg.

De Waal, A.G. and L.C.R.J. Willenborg, 1995, A view on statistical disclosure control of microdata. Paper submitted to Survey Methodology, Statistics Netherlands, Voorburg.

De Waal, A.G. and L.C.R.J. Willenborg, 1995, Global recodings and local suppressions in microdata sets. Report, Statistics Netherlands, Voorburg.

Pannekoek, J., 1989, The disclosure risk of bivariate population uniques (in Dutch). Internal note, Statistics Netherlands, Voorburg.

Pannekoek, J. and A.G. de Waal, 1994, Synthetic and combined estimators in statistical disclosure control. Report, Statistics Netherlands, Voorburg.

Pieters, A.J. and A.G. de Waal, 1995, A demonstration of ARGUS. Internal note, Statistics Netherlands, Voorburg.